Yocto Project ve Raspberry Pi 3 ile Gömülü Linux Geliştirme: Kapsamlı Bir Brifing

Bu belge, Yocto Project'in temel kavramlarını, Raspberry Pi 3 için özel bir Linux dağıtımı oluşturma sürecini ve bu süreçte karşılaşılan yaygın sorunların çözümlerini detaylı bir şekilde açıklamaktadır.

## 1. Yocto Project ve Gömülü Linux'un Temelleri

Gömülü Linux geliştirme, özel donanımlar için optimize edilmiş işletim sistemleri oluşturmayı içerir. "01\_10\_Introduction.pdf" kaynağına göre, bir Gömülü Linux sisteminin dört temel bileşeni şunlardır:

* **Toolchain (Derleme Araç Zinciri):** Hedef cihaz için kod oluşturmak için gereken derleyici ve diğer araçlar. Her şey Toolchain'e bağlıdır.
* **Bootloader (Önyükleyici):** Kartı başlatan ve Linux Kernel'ini yükleyen programdır.
* **Kernel (Çekirdek):** Sistemin kalbidir; sistem kaynaklarını yönetir ve donanımla arayüz oluşturur.
* **Root File System (Kök Dosya Sistemi):** Kernel başlatmayı tamamladıktan sonra çalışan kütüphaneleri ve programları içerir.

**Yocto Project Nedir?**

"01\_10\_Introduction.pdf" kaynağı Yocto Project'i şu şekilde tanımlar: "Yocto Project, geliştiricilerin herhangi bir donanım mimarisi için kendi özel Linux dağıtımını oluşturmalarına yardımcı olmak için açık kaynaklı, yüksek kaliteli altyapı ve araçlar sağlar." 2010 yılında iş tekrarını azaltma, kaynak sağlama ve kullanıcılar için bilgi sunma çabasıyla kurulmuştur. Birçok donanım üreticisi, açık kaynak işletim sistemi satıcısı ve elektronik şirketinin işbirliğiyle Linux Foundation'ın bir çalışma grubudur.

**Yocto Project'in Girdileri ve Çıktıları:**

Yocto Project'in çıktısı, "Linux Tabanlı Gömülü Ürün"dür. Bu ürün; Linux Kernel, Root File System, Bootloader, Device Tree ve Toolchain gibi bileşenleri içerir. Girdileri ise "ne istediğimizi tanımlayan veri kümesi"dir (Kernel Yapılandırması, Donanım Adı, Kurulacak Paketler/İkilikler).

**Temel Kavramlar:**

* **Poky:** Yocto Project'in "referans dağıtımı"dır; "örnek" olarak kullanılır. Kernel, Sistem ve uygulama yazılımlarını hedef donanım için oluşturmak üzere Bitbake, OpenEmbedded Core, meta-yocto-bsp ve dokümantasyonu içeren birleşik bir depodur. İkili dosyalar içermez, özel bir Linux dağıtımının kaynaktan nasıl oluşturulacağına dair çalışan bir örnektir. Yocto, organizasyonu ifade ederken, Poky indirilen gerçek kod parçalarını ifade eder.
* **Metadata (Meta Veri):** Yocto dünyasında meta veri, "derleme talimatları", "komutlar" ve "veri" anlamına gelir. Hangi yazılım sürümlerinin kullanıldığını, nereden alındığını ve yazılımın kendisine yapılan yamaları veya özelleştirmeleri belirtmek için kullanılır. Yapılandırma dosyaları (.conf), tarifler (.bb ve .bbappend), sınıflar (.bbclass) ve include dosyaları (.inc) koleksiyonudur.
* **OpenEmbedded Project:** "Sınıfının en iyisi bir çapraz derleme ortamı" sunar ve geliştiricilerin Gömülü Sistemler için eksiksiz bir Linux dağıtımı oluşturmasına olanak tanır. Yocto Project ve OpenEmbedded, "openembedded-core" adlı ortak bir meta veri koleksiyonunu paylaşır, ancak farklı odaklara sahiptirler. OpenEmbedded geniş bir mimari, özellik ve uygulama yelpazesi için kapsamlı meta veri sağlarken, Yocto Project güçlü, kullanımı kolay araçlar, meta veri ve Kart Destek Paketleri (BSP'ler) sunmaya odaklanır.
* **Bitbake:** Yocto Project'in temel bir bileşenidir. Make ile aynı işlevselliği gösterir. Python ve shell komut dosyası karışımı bir çekirdeği ayrıştıran bir görev zamanlayıcısıdır. Tarifleri okur ve paketleri getirerek, derleyerek ve sonuçları önyüklenebilir görüntülere dahil ederek takip eder.
* **meta-yocto-bsp:** Bir Kart Destek Paketi (BSP), belirli bir donanım cihazını, cihaz setini veya donanım platformunu desteklemeyi tanımlayan bilgi koleksiyonudur. Donanım özellikleri, kernel yapılandırma bilgileri ve ek donanım sürücülerini içerir. Poky'deki meta-yocto-bsp katmanı, BeagleBone, EdgeRouter ve genericx86 gibi çeşitli BSP'leri barındırır.

## 2. Raspberry Pi 3 ile Gömülü Linux İmajı Oluşturma Süreci

"Lab\_Manual\_1.pdf" ve "Sürdürülebilir Kentler ve Akıllı Teknolojiler" kaynakları, Raspberry Pi 3 için Yocto kullanarak bir Linux imajı oluşturma sürecini ve bu süreçte karşılaşılan hataların giderilmesini adım adım ele alır. Bu bölümde, "dunfell" Yocto sürümünü kullanarak elde edilen başarılı adımlar özetlenmiştir.

### 2.1. Geliştirme Ortamının Hazırlanması

* **Sistem Gereksinimleri:** En az 50 GB (önerilen 100 GB) boş disk alanı olan desteklenen bir Linux dağıtımı (Ubuntu 20.04 LTS gibi) ve Git, tar, Python gibi temel araçlar gereklidir.
* **Gerekli Paketlerin Kurulumu:** Geliştirme için gereken tüm paketler (gawk, wget, git, build-essential, python3, minicom vb.) sudo apt-get install ve sudo pip3 install komutlarıyla kurulmalıdır.

### 2.2. Yocto Kaynak Kodlarının İndirilmesi

* **Proje Klasörü Oluşturma:** ~/raspberryPi3/sources gibi bir ana proje klasörü oluşturulur.
* **Katmanları Klonlama:** dunfell dalından poky, meta-openembedded ve meta-raspberrypi katmanları git clone komutlarıyla indirilir. Bu adımda, dökümanlarda krogoth ve dunfell gibi farklı dalların olabileceği ve seçilen dala göre komutların değişmesi gerektiği belirtilmiştir.

### 2.3. Build Ortamının Yapılandırılması

* **Build Ortamını Başlatma:** cd ~/raspberryPi3/ dizininde source sources/poky/oe-init-build-env rpi-build komutu çalıştırılarak rpi-build adında bir çalışma dizini oluşturulur.
* **Katman Ekleme:** rpi-build dizininde bitbake-layers add-layer komutları kullanılarak indirilen meta-openembedded/meta-oe, meta-python, meta-multimedia, meta-networking ve meta-raspberrypi katmanları build projesine dahil edilir. Bu adımda, göreceli yol hatalarını düzeltmek için ../sources/ ön ekinin kullanılması gerektiği vurgulanmıştır.
* **local.conf Dosyasının Düzenlenmesi:** rpi-build/conf/local.conf dosyası, derlenecek makine (MACHINE ?= "raspberrypi3-64"), indirme (DL\_DIR) ve önbellek (SSTATE\_DIR) dizinleri gibi kritik ayarları içerir. Ayrıca, build süresini hızlandırmak için SSTATE\_MIRRORS ve indirme hatalarını önlemek için PREMIRRORS\_prepend gibi ayna sunucu ayarları eklenir. ssh-server-dropbear ve nano gibi ek özellikler EXTRA\_IMAGE\_FEATURES\_append ve CORE\_IMAGE\_EXTRA\_INSTALL\_append ile imaja dahil edilir. Seri port (UART) konsolunu aktif etmek için ENABLE\_UART = "1" satırı eklenmelidir.

### 2.4. Yaygın Hatalar ve Çözümleri

Gömülü Linux imajı oluşturma sürecinde birkaç yaygın hata ile karşılaşılmıştır:

* **meta-yocto Katmanı Hatası:** bblayers.conf dosyasında listelenen meta-yocto katmanının fiziksel olarak poky dizini içinde bulunmamasından kaynaklanır (özellikle farklı Yocto sürümleri arasında geçiş yapıldığında). Çözüm olarak, bblayers.conf dosyasından bu katmana yapılan referansın kaldırılması gerekmektedir.
* **do\_fetch Hataları (İndirme Başarısızlığı):** Yocto'nun gerekli kaynak kodlarını internetten indirememesinden kaynaklanır. Geçici ağ kesintileri veya kırık linkler bu duruma yol açabilir.
* **Ağ Bağlantı Sorunları:** ping komutlarıyla internet bağlantısı ve DNS çözümlemesi test edilir.
* **Kırık Kaynak Kodu Linki:** bmap-tools-native gibi paketlerin eski Yocto sürümlerinde (dunfell) geçerli olmayan linklere işaret etmesi sıkça görülür.
* **Ayna Sunucu Ekleme:** local.conf dosyasına PREMIRRORS\_prepend ayarıyla resmi OpenEmbedded ayna sunucuları eklenerek Bitbake'in orijinal bozuk link yerine alternatif kaynakları kontrol etmesi sağlanır.
* **Manuel İndirme ve Tarif Yamalama (.bbappend):** En inatçı durumlarda, sorunlu kaynak kodu manuel olarak indirip ~/yocto-downloads klasörüne taşımak ve bmap-tools\_3.5.bbappend dosyası oluşturarak Bitbake'e manuel olarak indirilen dosyanın yolunu ve doğrulama (sha256sum) kodunu bildirmek gerekmiştir. Bu yama dosyası, PV, SRCREV, SRC\_URI ve SRC\_URI[sha256sum] değişkenlerini override ederek kırık tarifi düzeltir.
* **do\_compile Hataları (setup.py bulunamadı):** İndirilen kaynak kodunun içindeki dosyaların beklenenden farklı bir alt klasörde bulunmasından kaynaklanır (setup.py dosyası bulunamaması gibi). Çözüm olarak, .bbappend dosyasına S = "${WORKDIR}/${BPN}-${PV}" satırı eklenerek Yocto'ya derleme işlemine hangi alt klasörden başlaması gerektiği belirtilir.
* **Checksum mismatch (Sağlama Toplamı Uyuşmazlığı):** İndirilen dosyanın beklenen güvenlik kodu (sha256sum) ile uyuşmaması durumudur. Yocto, bu hatanın çözümünü loglarda kendisi sunar ve SRC\_URI[sha256sum] satırının logda belirtilen doğru kodla güncellenmesi gerekir.
* **bitbake: command not found:** Yocto build ortamı her yeni terminal oturumunda veya sanal makine yeniden başlatıldığında tekrar etkinleştirilmelidir. source sources/poky/oe-init-build-env rpi-build komutu çalıştırılarak Bitbake ve diğer Yocto araçları terminale tanıtılır.
* **Read-only file system (Salt Okunur Dosya Sistemi):** SD karta imaj yazdırılırken sıkça karşılaşılan bir Linux sorunudur. Genellikle SD kart adaptöründeki fiziksel bir kilit düğmesinden veya SD kartın sistem tarafından otomatik olarak bağlanmış (mount edilmiş) bölümlerinden kaynaklanır. Çözüm olarak, kilit düğmesinin kontrol edilmesi ve sudo umount /dev/sdX\* komutuyla kartın tüm bölümlerinin sistemden ayrılması gerekir.

### 2.5. İmajın Oluşturulması ve SD Karta Yazdırılması

* **İmajı Derleme:** bitbake core-image-base komutuyla Linux imajı derlenir. Bu işlem, bilgisayarın performansına ve internet bağlantısına bağlı olarak uzun sürebilir.
* **İmaj Dosyasını Bulma:** Başarılı bir build sonrası imaj dosyası ~/raspberryPi3/rpi-build/tmp/deploy/images/raspberrypi3-64/ dizininde bulunur (core-image-base-raspberrypi3-64.wic.bz2).

1. **İmajı SD Karta Yazdırma:**bzip2 -d core-image-base-raspberrypi3-64.wic.bz2 komutuyla sıkıştırılmış imaj açılır.
2. sudo fdisk -l ile SD kartın cihaz adı (/dev/sdX) belirlenir.
3. sudo dd bs=4M if=core-image-base-raspberrypi3-64.wic of=/dev/sdX status=progress conv=fsync komutuyla imaj SD karta yazdırılır. **Yanlış disk adı seçimi veri kaybına yol açabileceği için bu adımda çok dikkatli olunmalıdır.**

## 3. Raspberry Pi 3 ile İletişim

### 3.1. USB-to-TTL Seri Port Bağlantısı

* **Jumper Ayarı:** USB-to-TTL dönüştürücünün voltaj jumper'ı **kesinlikle 3.3V** olarak ayarlanmalıdır. Raspberry Pi'nin GPIO pinleri 3.3V mantık seviyesiyle çalıştığı için 5V sinyal uygulamak karta kalıcı zarar verebilir.
* **Bağlantı Şeması:**GND (Dönüştürücü) → Ground (Raspberry Pi, Pin 6)
* RX (Dönüştürücü) → TXD0 (Raspberry Pi, GPIO14, Pin 8)
* TX (Dönüştürücü) → RXD0 (Raspberry Pi, GPIO15, Pin 10)
* Dönüştürücüdeki VCC pini Raspberry Pi'ye bağlanmamalıdır.

1. **Minicom Yapılandırması:**USB-to-TTL dönüştürücü bilgisayara takılır ve sanal makineye yönlendirilir (VirtualBox > Aygıtlar > USB > FTDI FT232R USB UART).
2. dmesg -wH komutuyla dönüştürücünün gerçek cihaz adı (/dev/ttyUSB0 veya /dev/ttyUSB1 gibi) tespit edilir.
3. sudo minicom -s komutuyla minicom ayar moduna girilir.
4. "Serial port setup" menüsünde "A - Serial Device" /dev/ttyUSB0 (veya bulunan ad) olarak ayarlanır.
5. "F - Hardware Flow Control" "No" olarak ayarlanır.
6. Ayarlar "Save setup as dfl" ile kaydedilir.

* **Raspberry Pi'yi Gözlemleme:** sudo minicom komutu çalıştırılır, Raspberry Pi'ye güç verilir ve açılış mesajları terminalde gözlemlenir. Giriş için kullanıcı adı root ve boş şifre kullanılır.

### 3.2. Ethernet (SSH) Bağlantısı

Derlenen Yocto imajı, ssh-server-dropbear özelliği sayesinde SSH sunucusu kurulu olarak gelir.

* **PC Ağ Ayarları:** Bilgisayarın Ethernet bağlantısı manuel (statik) IP yapılandırması ile ayarlanır: IP Adresi: 192.168.1.1, Netmask: 255.255.255.0, Gateway: 192.168.1.1.
* **Raspberry Pi Ağ Ayarları:** Derlenen imajdaki /etc/network/interfaces dosyası, eth0 arayüzü için statik IP (192.168.1.2) ile zaten yapılandırılmıştır.
* **Bağlantı Testi:** Raspberry Pi ve bilgisayar Ethernet kablosuyla bağlanır. ping 192.168.1.2 komutuyla bağlantı test edilir.
* **SSH ile Bağlanma:** ssh root@192.168.1.2 komutu kullanılarak Raspberry Pi'nin terminaline erişilir. root kullanıcısı için şifre istemeyecektir.

Bu detaylı brifing, Yocto Project ile Raspberry Pi 3 üzerinde gömülü Linux geliştirme sürecini, karşılaşılan sorunları ve çözümleri kapsamlı bir şekilde açıklamaktadır.